The Integration of Image Segmentation Maps Using Region and Edge Information

Chen-Chau Chu, *Member, IEEE,* and J. K. Aggarwal, *Fellow, IEEE*

***Abstract*— We present an algorithm that integrates multiple region segmentation maps and edge maps. It operates independently of image sources and specific region-segmentation or edge-detection techniques. User-specified weights and the arbitrary mixing of region/edge maps are allowed. The integration algorithm enables multiple edge detection/region segmentation modules to work in parallel as front ends. The solution procedure consists of three steps. A maximum likelihood estimator provides initial solutions to the positions of edge pixels from various inputs. An iterative procedure using only local information (without edge tracing) then minimizes the contour curvature. Finally, regions are merged to guarantee that each region is large and compact. The channel-resolution width controls the spatial scope of the initial estimation and contour smoothing to facilitate multiscale processing. Experimental results are demonstrated using data from different types of sensors and processing techniques. The results show an improvement over individual inputs and a strong resemblance to human-generated segmentation.**

***Index Terms*— Contour smoothing, information integration, maximum likelihood estimation, multiscale processing, segmentation.**

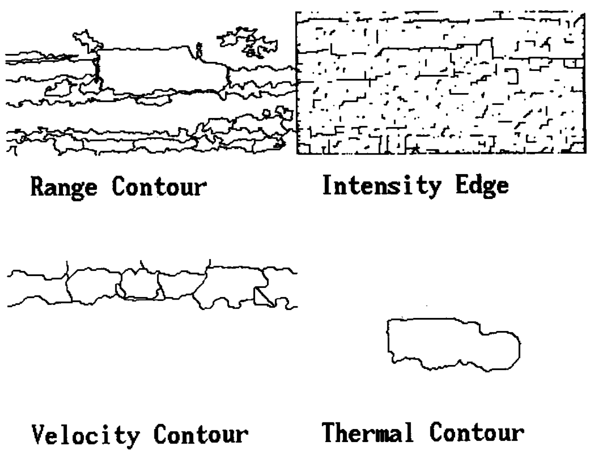


Fig. 1. Four source images

I. Introduction

A GOOD segmentation is vital to various image analysis tasks. However, a good image segmentation is not always achievable by using a single technique across a wide range of sensing modalities and image contents. When different segmentation techniques are applied to images acquired through multiple sensing modalities (Fig. 1), different segmentation maps are generated (Fig. 2). One has to resolve the differences between all such segmentation maps to benefit from the rich information provided by various sources. Information integration is a suitable approach to enhance system performance by verifying cues from one source to another. It is also necessary because the information loss during the image acquisition process is significant. The consistent information from different sources should reinforce one another while the contradictory parts of the information, either noise or errors, should attenuate one another. Essentially, the signal-to-noise ratio is enhanced through information integration. Improved results in image segmentation and analysis due to information integration are reported in [1]–[5].

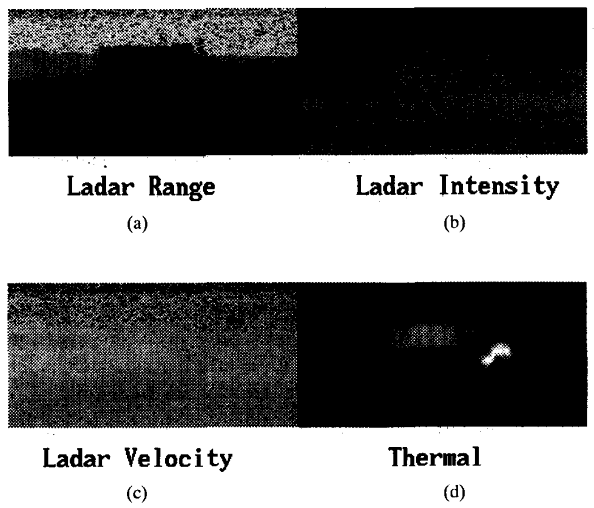


Fig. 2. Four input segmentation or edge maps.

In this paper, we report an algorithm developed to integrate both region and edge information without the intervention of high-level knowledge. These segmentations may be derived

Manuscript received February 7, 1991; revised January 21, 1993. This work was supported in part by the DoD Joint Service Electronics Program through the Air Force Office of Scientific Research (AFSC) under Contract F49620-89-C-0044, and in part by the Army Research Office under Contract DAAL03–87–K–0089 and DAAL03–91–G–0050. Recommended for acceptance by Associate Editor M. Levine.

C.-C. Chu was with the Computer and Vision Research Center, Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712–1084. He is now with Schlumberger Austin Systems Center, Austin, TX 78720–0015.

J. K. Aggarwal is with the Computer and Vision Research Center, Department of Electrical and Computer Engineering University of Texas, Austin, TX 78712-1084.

IEEE Log Number 9213202.

from different sensing modalities, segmentation techniques, and control parameters. The developed algorithm operates independently of the problem domains, the segmentation techniques, and any combination of edge and region maps. The integration algorithm generates a *consensus* of the *true* underlying segmentation from multiple observations. Therefore, the basic task of the proposed integration algorithm is *estimation.* Since the original images (Fig. 1) are not required during the operation, various segmentation techniques can work in parallel as the front end of the integration module, and the segmentation modules can work with the integration module in a pipeline fashion. Our algorithm achieves promising results based on the above-mentioned criteria. Furthermore, it allows the flexibility of user-specified weights on different information sources since information from different channels is not equally reliable. Early results of this algorithm were reported in [6].

Section II reviews related literature, and Section III provides the formulation of the underlying problem as an optimization problem. Sections IV, V, and VI address the algorithm design after the original problem is decomposed into three subproblems. Section VII reports the implementation and the results, and compares the proposed algorithm with similar works. Finally, Section VIII summarizes the work.

II. Related Works

Most research on the subject of integrating edge and region information falls into four categories: 1) techniques using high-level knowledge, 2) algorithms based on pixel-wise Boolean-logic operations, 3) algorithms for specific imaging modalities or segmentation techniques, and 4) approaches using *a priori* information and probabilistic models. High-level knowledge is applied to segmentation [7] and shape extraction [8] with some success. Research literature has not discussed knowledge-guided integration of multiple segmentation maps. However, a high-level approach to region/edge integration has to assume some knowledge of the contents of the images, and usually becomes domain-dependent for this reason. In this paper, we do not consider the knowledge-based approach. The paper does not consider the fusion of gray level information, such as that in [9], since it does not provide cues for segmentation.

Pixel-wise Boolean-logic operation as a strategy of integration is computationally inexpensive and easy to implement. In [1], pixel-wise Boolean-AND operations are used to confirm target positions from laser radar and infrared images. Edge information is used together with region segmentation techniques in [10]. In [11], an edge operator is applied only at pixels on the zero-crossings contour of the Laplacian-of-Gaussian operation result to save computation. However, using the AND operator usually generates fragmented edge patterns, while using the OR operator tends to generate spurious edge patterns. In general, the more segmentation maps are integrated at one time, the worse the results become. Besides, this approach cannot address such important issues as edge contour connectivity, contour smoothness, and the weights on different information sources.

The integration algorithms designed for specific segmentation algorithms are usually effective because of the closer cooperation between the segmentation and the integration modules. Edge and gray-level information from infrared images are used jointly for target extraction in [2]. Some researchers [12]–[14] use edge detection techniques to refine the results of region segmentation, or vice versa. In [12], the edge detection threshold is adjusted to generate a prefixed number of regions in a feedback control loop. In [13], images are segmented by *split-and-merge* based on *quadtree decomposition.* The second stage uses objective function to detect and remove artifacts generated by the quadtree segmentation method. In [14], edge information is used in a relaxation format to refine region segmentation generated by partitioning the co-occurrence matrix. These algorithms [2], [12]–[14] require the presence of the original images. In addition, these algorithms have strong links to specific segmentation algorithms and the corresponding data structures. Strictly speaking, these algorithms are segmentation algorithms using more than one processing technique instead of the independent integration algorithms that can use information from front-end modules. Besides, these algorithms do not consider multiple segmentation maps as input, let alone allow weights on the input.

The probabilistic approach usually has a neat mathematical formulation and provable optimality. References [15] and [16] deal with the information fusion problem using the classical estimation theory instead of working directly on regions and edges. Observations from multiple sensors are used to set up a transcendental equation. The equation is then solved iteratively with the knowledge of the covariance matrix of the input signal. However, the contribution of a sensor is accepted or rejected in an *all-or-none* fashion in [15] rather than incorporated in a weighted average fashion. In addition, it is unclear how the techniques in [15] and [16] are applicable to the problem of integrating edge and region information. No result is present in [15] and [16] . A Bayesian technique to combine information from edges is proposed in [17]. This technique assumes *a priori* information. The result is a map of *edge-likelihood* measures without considering edge connectivity. However, real-world images never match well to closed-form probability distributions. The *a priori* information is difficult to obtain, if not impossible. None of the three works considers contour continuity and smoothness; nor do they consider region size and compactness. The inability to account for these factors is a serious disadvantage for any integration algorithm used for region-based image analysis.

The proposed algorithm differs from all of the four approaches cited above in computational strategies, flexibility, and controllability. The proposed algorithm operates independently of sensing modalities and processing techniques. It allows user-specified weights to emphasize more reliable information sources. A user can select a proper *fusion radius* to dictate the scope of information fusion. Both region and edge maps are allowed in its operation. Contour connectivity and smoothing are given proper attention such that the output of this integration module can be used by high-level interpretation-oriented modules.

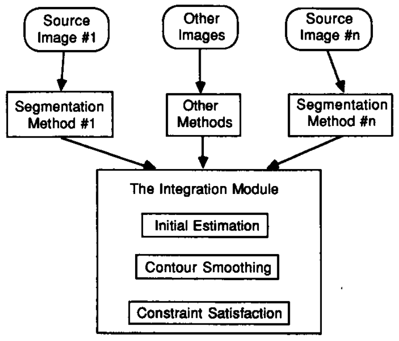


Fig. 3. System overview.

III. Problem Formulation

It is assumed that a *true* region contour map exists as the signal source. The signal is contaminated by various noise sources of different characteristics during the process of image acquisition, preprocessing, and segmentation. The objective of the integration module is to recover the original contour map from multiple contaminated copies using minimal knowledge about the signal and noise sources. Section III-A summarizes the architecture and features of the proposed integration algorithm, whereas Section III-B presents the operational strategy of the algorithm.

This work uses only the *contour* (position and length), the *size*, and the *neighboring relationship* attributes of regions. We do not use other *regional information* because that would require the integration module to know about the construction of the front-end (region-growing-based) segmentation algorithms and about their operational results before a contour is generated. If a region-based segmentation algorithm can provide a contrast measure (e.g., difference in average intensity between regions A and B), then this information can be incorporated as edge strength, too. From another perspective, only the region information related to edges can be integrated with edge information. Apparently, not all attributes of regions can be utilized in this manner.

A. System Paradigm, Features, and Requirements

The proposed integration algorithm generates an edge map from all input segmentation maps. Fig. 3 shows an overview of the entire algorithm. The resultant edge map may then be converted to a region segmentation map considering only closed edge contours if needed. Additional constraints can be applied to the properties of the resultant regions. The integration process before the final output, however, is independent of a particular domain. Since the output of this integration module is to be used by a region-based image interpretation module [4], [5], we are more concerned with the properties of regions.

The input data sets are assumed to have the following properties: 1) all source images are *registered* (i.e., they cover the same solid angle) before they are processed. However, how they are generated is *not* known. The original images are *not* used in the integration process. 2) All pixels on region contours are assumed to have an edge-strength value of 1.0. The edge strengths for noncontour pixels are zero. All pixels in an edge map have a strength in the range of [0.0, 1.0]. Edge strength is considered as the precision of observations. The input may contain edge segments that are not closed. Hereafter, we consider all the input maps as edge maps without losing generality. 3) The observation noise sources are independent, zero-mean, additive Gaussian. The noise processes are independent from channel to channel and from pixel to pixel. In reality, there is no definitive evidence that the noise sources are Gaussian. However, the Central Limit Theorem indicates that the accumulated effects of various noise sources, all with unknown characteristics, are best modeled by a Gaussian source. 4) No other *a priori* information is available, and no higher-level (semantic) knowledge is used.

The output should satisfy the following properties: 1) the output is a *negotiated* result from the input data set (the *fidelity criterion*) rather than a *selection* from them. The scope of the data fusion should be controlled by the fusion radius. 2) The region contours are smoothed (the *smoothness criterion*) subject to the fusion radius. 3) The edge contour generated as output is single pixel wide, without redundant contour connectivity. 4) Edge pixels connected in the input sets should remain connected in the output set (the *connectivity constraint*). 5) All regions are larger than a size threshold (the *region-size constraint*). 6) All regions are more compact than a compactness threshold (the *contour-compactness constraint*). The compactness is defined as the square of region contour length divided by region area. Constraints 5 and 6 are posted by the image interpretation system that uses the integration result. Only the constraints on region size and compactness are considered in this work. However, various constraints can be specified depending on the problem domains and the processing needs after the integration (Section VI). Both the initial estimation of contours and the curve smoothing stage should use only local information and should be controlled by a spatial parameter.

Several control parameters provide the necessary flexibility for the system. 1) The *fusion radius* (size of local windows) λ is specified in pixels. The role of λ is similar to that of σ as the *channel width* in the *multiple channel resolution* concept [18]. 2) The weighting factors for each map are given, and the weights are applied to all pixels in the associated input maps. If weights on the individual pixel positions are given (as a map of weights), then the weights are applied pixel by pixel. The paper is not concerned with how the weights are determined. 3) The thresholds on region size and compactness are determined by the application context.

B. Strategy: Multistage Optimization

Based on the specifications on input/output conditions, the region/edge integration problem is decomposed into three major subproblems. The first subproblem (estimation) is to estimate the *true* region contours from given multiple observations. The second subproblem involves region contour smoothing. The third subproblem (constraint satisfaction) is to satisfy additional nonnegotiable constraints on integration output according to the application context. Since the three subproblems are related, an ideal solution should formulate the three as a single problem.

Therefore, the integration problem is formulated as: *Given several sets of edge pixels and associated weights, solve for another set of edge pixels representing all input sets and exhibiting certain properties.* Let I denote the set of all input edge pixels and O denote the set of output edge pixels. In addition, let $C_e(\cdot)$ denote the cost function of estimation error for a single input edge pixel, and denote $C_s(\cdot)$ the smoothing cost for an output edge pixel. Based on the kind of constraints posted in the above-mentioned context, an objective function to be minimized can be described in the form of

$$\sum_{\forall T_i \in I} C_e(T_i) + \sum_{\forall Q_j \in O} C_s(Q_j) \tag{1}$$

subject to such additional constraints as contour connectivity, region size, region compactness, single-width contour in the solution set, etc. The last several constraints are nonnegotiable. Therefore, they are not contained in the summation of the estimation cost and the smoothing cost. A complete specification of (1) is very tedious and hard to describe in a pure symbolic form.

The first term in (1) is a *fidelity* term. It ensures that the solution is a *negotiated* result from all the inputs, and the output is a sufficient representative of the input data. $C_e(T_i)$ is the estimation *cost* due to an input edge pixel T_i. Some criterion has to be chosen before optimization begins. The second term in (1) is a *smoothness* term. It minimizes the jaggedness of the resultant edge patterns to favor long straight edges. As a result, a minimal number of the edge pixels in the output set are preferred. $C_s(Q_j)$ is the cost of curve smoothness at a point Q_j and is usually defined as $\kappa^2(Q_j)$ in the continuous case, where κ is the local curvature (in [13]) or high-order derivatives (in [19]). In the discrete case, some complications exist (Section V).

The objective function in (1) is difficult to optimize analytically for two reasons. First, the number of pixels in the solution set is unknown or changing until the integration is complete. Therefore, the formulation in (1) is a mixed detection–estimation problem. Second, the additional constraints on contour connectivity and region properties are unnegotiable constraints that *must* be satisfied. These factors introduce strong nonlinearity into the system and prevent a clean analytical solution. One can certainly quantify the constraints on region size, contour compactness, etc., into additional terms in (1), but that would not make the function easier to optimize.

Consequently, we propose a solution procedure that decomposes the original formulation into three stages to pursue a *suboptimal* solution. The first stage is an estimator that generates an initial solution and minimizes the *fidelity* term (Section IV). The second stage employs a potential-energy model to minimize the *smoothness* term iteratively (Section V). The third stage checks the unnegotiable constraints and merges regions that violate such constraints (Section VI). There are different ways to pursue a multistage, suboptimal solution. The authors do not claim to have found the best one, however.

IV. The Initial Estimation

The objective of the initial estimation stage is to estimate the *true* positions of edge contours from multiple observations while maintaining the estimation results in connected contours. All region segmentation maps are converted to edge maps for operational purpose. This study chooses the *maximum likelihood estimation* (MLE) [20], [21] as the estimation strategy because it requires minimal *a priori* information and easily incorporates user-supplied weights on information sources. Therefore, we are trying to minimize

$$\sum_{\forall T_i \in I} C_e(T_i, P_i) \tag{2}$$

with additional constraints on a) contour connectivity (Section IV-B) and b) single-pixel output contour. T_i (the observation) is assumed to be a contaminated version of P_i. It is not unusual that P_i may be equal to P_j when $i \neq j$.

For the operational purpose, mle is reduced to a weighted average format under suitable assumptions (refer to the Appendix). In other words, for a set of independent observations z_i's, the mle estimate of the edge element s is given by

$$s_{MLE} = \frac{\sum_{i=1}^{m}(1/\sigma_i^2) z_i}{\sum_{i=1}^{m}(1/\sigma_i^2)}. \tag{3}$$

Note that the mle result s_{MLE} derived in (3) also minimizes

$$\sum_{i=1}^{i=n} \frac{(z_i - s)^2}{\sigma_i^2}. \tag{4}$$

The exact format of the cost function $C_e(\cdot)$ becomes less important because the optimal estimators for a wide class of cost functions, under proper assumptions, are eventually the same [20], [21].

The mle process is a *point* process. It has no built-in ability to deal with the contour connectivity between edge pixels. It is possible that the mle solutions for two connecting edge pixels are disconnected. Therefore, a pixel-by-pixel mle is used to estimate the positions of edge elements as a first step (Section IV-A). Because the estimation process for each edge pixel is independently computed, a parallel implementation is possible. The contour connectivity is enforced in the second step (Section IV-B).

The formulation in (2) is similar to those of clustering and vector quantization in the sense that the number of elements in the output set is not fixed. The difference is that we require that P_i be connected to P_j if T_i is connected to T_j (the *connectivity* constraint). If P_i and P_j are not connected when they should be, then this constraint will add in edge pixels to connect P_i and P_j. The edge pixels added for this purpose do not have a direct correspondence to input edge pixels or their mle results.

A. The Search Circle

The mle formulation and computation are straightforward once all the samples for an observed variable have been

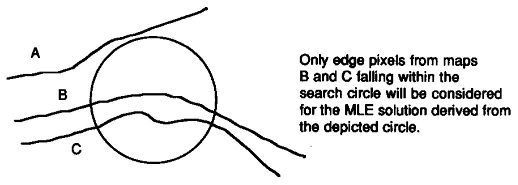

A
B
C
Only edge pixels from maps B and C falling within the search circle will be considered for the MLE solution derived from the depicted circle.

Fig. 4. Search circle determines the scope of sample collection.

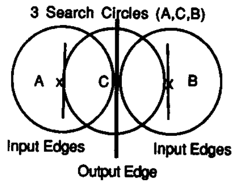


Fig. 5. The decision among overlapping decision circles.

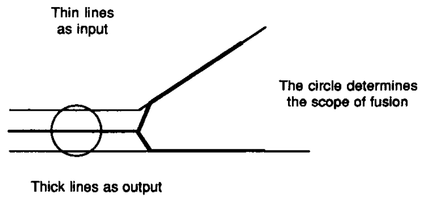


Fig. 6. Connectivity between mle solutions.

determined. It is necessary but sometimes difficult to determine which samples (edge pixels on input maps) should be considered in the estimation for a particular output (edge pixels on the output map). The difficulty is that no *a priori* information exists to determine the scope of the sample collection. Cross-validation [19] is computationally expensive. Including all samples does not seem to be reasonable, especially for large images. The desirable approach should employ some sort of local consistency or clustering-based data localization.

The authors propose to use a user-selectable parameter, λ, the *fusion radius*, to determine the radius of a *search circle* (SC). Samples located within a search circle are considered as *relevant* for a potential *true* edge pixel. Within each SC, for every input set i, the edge pixel from the input edge map i closest to the center of the SC is collected into a sample set (Fig. 4). If more than one edge pixel from any set i satisfies the criterion of *the shortest distance*, then all of them are collected as observation samples. Using a larger λ allows the merger of edge contours with larger distances between them. In addition, smoother edge patterns are generated because overlapping SC's contain mostly the same samples. After the mle solution is found from a sample set (by (3)), a verification is performed. Any sample edge pixels more than λ away from the mle solution are removed from the set. If there is such a removal, the mle solution is recalculated until no more samples are rejected.

Note that an SC does not always have an edge pixel at its center. In addition, the mle solution derived from an SC may not always be an edge pixel in the input set. Therefore, the mle result is a *negotiation* among all samples rather than a *selection* among all the samples. Note that each input edge pixel may be involved in multiple SC's. In other words, for each $T_i \in I$, there are multiple candidates for its uncontaminated value P_i. The SC that involves the most samples (or highest total edge strength if weighting is used) is selected. This mechanism maps each input edge pixel T_i to the mle solution close to T_i (within λ) and supported by the largest (or heaviest) local sample set. Thus, both local consistency and channel resolution are incorporated into the solution.

Consequently, neighboring edge segments with a distance of less than 2λ (likely to be generated from a single contour by different noise processes) would generate a resultant edge located in between the two input edge segments. In Fig. 5, the result from the SC centered at C overrides the results from SC's A and B centered at the two edge pixels P_1 and P_2, respectively. Note that SC C collects samples from both input edges while A and B collect samples from only one of the input edges. The result from the SC at C dominates even though the SC at A is processed first. All subsequent references to the SC's A and B, as well as to the mle solutions determined by A and B, are mapped to the SC at point C and its mle solution. In comparison, the mle solutions generated by SC's A and B (and SC's centered along the two input edges) will not appear in the output set.

B. Connectivity

After the pixel-by-pixel mle is performed, the resultant edge pixels in the solution set may *not* be fully connected. The gaps usually occur at junction points where two or more edge segments join (Fig. 6). Among many different ways to deal with the situation, the key is to preserve the junction structure. A heuristic approach is to connect every isolated pixel to its nearest neighbor. However, this does not guarantee that connected edge pixels will generate connected mle results. Besides, the connectivity in the input segmentation maps is not consulted as it should be.

The proposed algorithm enforces a linear connection between the mle solutions generated by two SC's if a) the centers of two SC's are adjacent, and b) the centers of both are edge pixels in the input maps. Sometimes there is more than one candidate to enforce the linear connectivity. The selection rules operate in the following order of priority: a) The result from an SC containing the largest total weight on samples is selected. Thus, the connectivity in a heavier weighted input dominates the edge connectivity in the output. b) The centroid of the mappings of all neighboring SC's is calculated if none of them dominate. The SC whose mapping is closer to this centroid is selected. The principle behind these rules is to respect observation precision (in the form of weights) and local consistency.

C. Procedure

1) *Edge Map Preparation*—All segmentation maps are converted into edge maps. Region contours are extracted as edges while edge information remains in its original format.

2) *Edge Composition*—The composite edge map is generated from all input as $E_c(x,y) = \sum_{i=1}^{i=m} w_i(x,y)e_i(x,y)$. The e_i's are edge strengths derived from input map i. The *edge pixel connectivity* is defined based on this composite edge map rather than on the individual input map. The w_i's are the given weights on individual data channels.
3) *Weak Edge Removal*—The composite edge map can be thresholded to remove very weak edge pixels. This step accelerates the overall operation speed by reducing the number of edge pixels to be processed in later stages. However, this step is not essential to the algorithm, and it is not used in the examples shown later.
4) *mle operation*—Solve for the mle solutions for all SC's that cover at least one edge pixel from the input maps. This step can be further decomposed into several steps. Operations within one SC are independent of operations in other SC's. Therefore, parallel processing can greatly improve the performance at this stage.

    a) Generate one SC for every pixel position in the image plane. Process only those carrying at least one edge pixel.
    b) Collect samples from each SC. Compute the ML estimation as a weighted average. Verify the samples (within the current SC) against the mle solution. Drop the samples more than λ away from the mle solution, and update the mle result if necessary.
    c) Record the mapping toward the mle solution for each contributing sample. When an edge pixel receives different mappings from multiple SC's, choose the SC carrying the heaviest weight.

5) *Neighbor connectivity*—The $T_i - P_i$ mappings are examined and neighboring edge pixels are connected linearly. If extra edge pixels have to be filled between two mle solutions, then the edge strengths of these pixels are linearly interpolated between the edge strengths of the two terminal edge pixels. If a certain pixel position receives multiple edge strength values, the strongest one is used.
6) *Postprocessing*—Thin the edge segments to one-pixel wide. However, if the edge strength of a pixel is the local maximum in a 3×3 neighborhood, then that pixel is not removed. Therefore, the edges are thinned, but the thinning is guided by edge pixels with strong values (reliable observations).

V. Region Contour Smoothing

At the end of the mle stage, an edge map of edge pixels P_i is obtained. The objective of the region contour smoothing stage is to reduce curvature directly. As a result, the total contour length is reduced indirectly. However, the need to smooth the region contour must coexist with the need for a precise estimation. The contour smoothing task (the second term in (1)) is now defined as:

$$\begin{array}{rl} \text{Given} & \text{A set of edge pixels}\{P_j\} \\ \text{Solve for} & \text{A set of edge pixels } \{Q_j\} \\ \text{Minimize} & \sum_{j=1}^{j=m} C_s(Q_j) \\ \text{subject to} & (a) dist(P_j, Q_j) \le \rho \text{ for all } j, \\ \text{and} & (b) dist^2(Q_i, Q_j) \le 2 \text{ if } dist^2(P_i, P_j) \le 2. \end{array} \tag{5}$$

$dist(\cdot)$ is the usual Euclidean distance measure defined on integer coordinates. The constant 2 is used to represent the 8-connectivity. The *finite-drift* constraint (a) stipulates that no pixel may move away from its original position (as determined by the mle stage) by more than ρ. The *connected-neighbor* constraint (b) stipulates that neighboring edge pixels must remain neighbors. In this work, we set $\rho = \lambda$, but ρ and λ may be set to different values. Since all samples (edge pixels in input edge maps) contributing to the associated mle solution are within λ of the mle solution, it is unreasonable to set $\rho > \lambda$. A larger ρ results in smoother curves because it allows more leeway for an edge pixel to adjust its position to reduce local curvature. One may set different drift limits on each edge pixel. However, preliminary results show that only minor differences exist in the results.

$C_s(\cdot)$ is a cost measure that depends on the local curvature measurement κ at the current position of P_i. In the analog case, $C_s(\cdot)$ is often defined as $(\partial\psi/\partial t)^2$ where t is the edge length variable tracing along the edge contour and ψ is the orientation of the edge. Since the image is discrete, $\partial\psi/\partial t$ can be approximated by θ_i (an angle). θ_i is the measurement of the orientation *change* between $\overline{P_iP}$ and $\overline{PP_{i+1}}$ for edge pixels P_i's in the 3×3 neighborhood of point P. Therefore, one might expect to define $C_s(P)$ as

$$C_s(P) = \sum_{i=1}^{i=k} \theta_i^2, \tag{6}$$

where k is the number of connected neighbors of the central pixel P. The optimal solution in this formulation is the point that results in $\theta_1 = \theta_2 = \ldots = \theta_k$. However, there is no analytical solution to find such a point, especially when the two additional constraints in (5) are applied. Therefore, this approach is considered as inoperable and hence abandoned.

A. Iterative Optimization

In response to the above concerns, we have designed an iterative optimization technique. A *spring-and-node* model is chosen to solve the contour smoothing problem through iteration. At the start of the smoothing stage, each edge pixel is represented as a node. Neighboring nodes are connected by springs. The driving forces are generated by the springs, and they push the nodes into positions that minimize the total potential energy stored within the springs (Fig. 7). The *connected-neighbor* constraint and the *finite-drift* constraint act as additional springs to restrict the motion of a pixel. Then the nodes are *released* from their initial positions to move freely subject to the spring forces.

The springs will force the central node to move to the centroid of all its connected neighbors such that the net force is zero and the stored potential energy is also minimized. Note

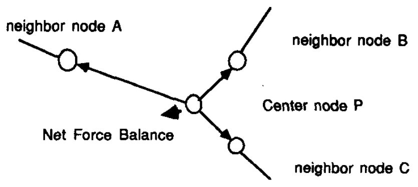


Fig. 7. Curve smoothing at a junction point.

that the centroid of all neighbors is also the MMSE solution of minimizing $\sum_{i=1}^{i=k} dist^2(P_c, P_i)$. When $k = 2$ (no contour junction), the *centroid solution* guarantees that the three nodes are collinear and also minimizes the energy defined in (6).

Therefore, the definition of $C_s(\cdot)$ must consider more than two neighbors. In addition, it is preferred not to involve angles directly. The resultant function form can be considered as a generalization of the energy function used in the *Snake* [22, eq. (13)]). We define the resultant energy function for curve smoothing as

$$C_s(P_i) = E_{int}(P_i) = \alpha \sum_{j=1}^{j=k} |\mathbf{v}_i - \mathbf{v}_{i,j}|^2 + \beta \sum_{j=1}^{j=k} |\mathbf{v}_{i,j} + \mathbf{v}_{i,j+1} - 2\mathbf{v}_i|^2, \qquad (7)$$

where $\mathbf{v}_i$ is a vector form of (x_i, y_i) (the coordinates of P_i), and $\mathbf{v}_{i,j}$'s are the neighbors connected to $\mathbf{v}_i$. $\mathbf{v}_{i,k+1}$ is wrapped back to $\mathbf{v}_{i,1}$. Because every node is moving at the same time, the centroid solution has to be applied repetitively until it converges. The α_i's and β_i's at all points P_i's are assumed to be equal to α and β, respectively. The α term is proportional to the potential energy from the stretch of the springs. The β term is the smoothness term and corresponds to the bending energy of the springs. The *centroid solution*

$$P_c = \frac{1}{k} \sum_{j=1}^{j=k} \mathbf{v}_{i,j} \qquad (8)$$

minimizes both the α and β terms in (7). The fact is easily verified by taking the partial derivative of (7). When $k = 2$, the solution that minimizes $\sum_{i=1}^{i=n} E_{int}(i)$ also minimizes [22, eq (13)]. Note that [22] does not consider the cases of $k \geq 3$.

In general, the point P_θ that satisfies $\theta_1 = \theta_2 = \ldots = \theta_k$ is *not* the centroid of all the k connected neighbors. However, the centroid P_c and P_θ are usually so close that after the quantization to integer coordinates, P_c and P_θ almost always fall on the same coordinates. Besides, when the summation in (7) approaches its theoretical minimum, all the k neighboring nodes must be on a circle surrounding the central node with equal spanning angle—exactly the same geometric configuration required in (6).

B. Procedure

In each iteration, each node is moved to the centroid of its neighboring nodes with spring connections. A step size is determined as the displacement necessary to move an individual node to its target position. However, the step sizes must satisfy the two constraints as specified in (5). Otherwise, a smaller step size is chosen along the same orientation. The verification of step size is repeated until the determined step size is too small ($< \epsilon$) or until the two constraints are satisfied. The same operation procedure is applied to every pixel on the input edge map. The process is iterated until no pixel changes its integer (grid) coordinates, and the maximal drift of pixel positions is smaller than ϵ (set to 0.1 pixel in our implementation). The exceptions are the nodes originally on the image frame boundaries. These nodes are allowed to drift only along the image frame boundary. The purpose is to make sure that separate regions at the input stage remain separate during the contour smoothing stage. If two or more nodes are located at the same grid position, only one needs to be represented in the output. Since each node and its neighborhood can be operated independently, this stage also can take the advantage of parallel processing. One has to watch for oscillations, however. The procedure can be described in the following pseudo code:

```
**input** $P_j$; /* input data set: the mle output after thinning */
**repeat**
**begin**
    coordinates.moved := **false**;
    maximal.move := 0;
    **for all** $P_j$ **do**
    **begin**
        X := centroid.of.all.neighbors($P_j$);
        offset := X - $P_j$;
        offset := check.constraints(offset);
          /* finite-drift and connectivity */
        **if** (offset $\geq \epsilon$) **then**
        **begin**
            $P_j$ := $P_j$ + offset; /* moves $P_j$ to X */
            maximal.move := max(maximal.move, offset);
            **if** (the integer coordinates of $P_j$ are changed)
            **then** coordinates.moved := **true**;
        **end**
    **end**
**end**
**until** ((maximal.move $\leq \epsilon$) **and** (**not** coordinates.moved));
```

VI. Compactness And Size Constraints

After the contour smoothing stage, we have an edge map with smoothed edge segments which do not always form closed contours. If an edge map is the desired output, the algorithm stops here. Otherwise, the edge map is converted into a region map by *region growing* from non-edge pixels. Consequently, *dangling* edge segments are removed. Regions that do not satisfy the output specifications are merged with their neighbors. Two constraints are used in our implementation: a) the contour compactness constraint, and b) the region size constraint. The two constraints are essential in practical situations, since the interpretation of very small regions in an image is usually unreliable. These two thresholds are set according to the needs of the application domain.

The problem is to determine the neighbor with which to merge. One may consider the following criteria: the length

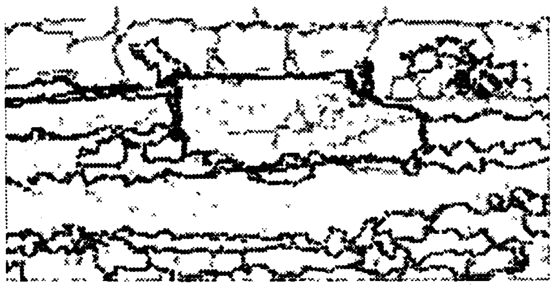


Fig. 8. The intermediate result after the mle stage; $\lambda = 2$ (pixels.)

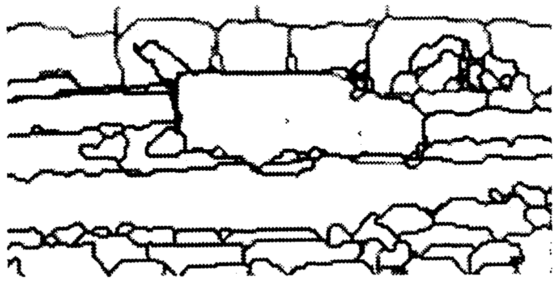


Fig. 9. The intermediate result after curve smoothing.

of the shared contour between two regions; the strength of the shared contour segments separating them; the ratio of the shared contour to the overall contour length, etc. The choice should be determined by the application context. A merit function is designed as

$$M(R_s, R_n) = \frac{\text{shared.contour.length}(R_s, R_n)}{\text{segment\_size}(R_n) \times \text{edge.strength.between}(R_s, R_n)}, \tag{9}$$

where R_s is the region to be eliminated, and R_n is the neighborhood region under merger consideration. The neighbor with the highest value of this objective function is chosen. In other words, the selection favors neighboring regions with long shared contours, small sizes, and weak separation strengths. The following summarizes the operation procedure for this stage:

1) *Region map generation*—Regions are built from nonedge pixels (whose edge strengths are zero). Optionally, very small regions (smaller than 10% of the region size threshold) are immediately marked off without merging into their neighbors to accelerate the merging process. The *holes* thus generated are covered in step 4. Information of the neighboring regions and of the neighborhood relationships is recorded so as to compute the merit function in (9).
2) *Detection of the violations*—Segments too small or not compact enough are marked for merger with their neighbors. Threshold values on minimal region size and compactness ratios have to be supplied by the user, or possibly by a high-level image analysis module. If additional constraints are desired, they are all applied at this stage to mark the regions to be eliminated.
3) *Selection of a merger partner*—Consider a region R_s to be eliminated. A search is conducted among all its neighboring regions to select an R_n that maximizes the merit function. In practice, a more complicated merit function or a multistage decision strategy can be used to select a suitable neighbor. Merge the two regions and update all data tables. Loop through steps 2 and 3 until no segment violates the constraints.
4) *Post-processing*—All regions are extended to their exterior until all pixels on the image plane are covered.

Note that this constraint satisfaction stage is devoted mostly to checking the properties of the resultant regions derived from the integration. The task of selecting two regions to merge is a well-studied topic in region-based segmentation. Therefore, many known criteria and attributes can be applied at this stage. The merit function in (9) can be designed to incorporate region-oriented attributes. For example, one can use the similarity between the average pixel values within two regions. However, such techniques are usually more context-dependent than the contour information. For example, one can fit planar surfaces to pixel values and then compare the direction of the (fitting) surface normals. Such a technique makes good sense for range images, but it is not very meaningful for thermal images. Obviously, the more specialized the merit function is, the less adaptable it will be toward other problem contexts. This is a typical design compromise one has to face.

VII. Implementation, Results, and Discussion

The algorithm is implemented using the C language on an IBM RT PC ($\approx$2.2 VAX-MIPS) running IBM AIX. The inputs are derived from real laser radar [23] images and corresponding thermal images. The ladar data we use have three registered data channels: range, intensity, and velocity. Fig. 1 presents the four source images for the first example. A single truck heading to the right is shown in a broadside view. Fig. 2(a)–(d) contains the following input maps, repsectively: the surface fitting-based segmentation of range data [3], the edge detection result from ladar intensity data, the statistics-based segmentation of ladar velocity data [3], and the threshold-based segmentation of thermal data. Each image has a size of 128 by 256. Note that each input segmentation map has its own problems and merits. Fig. 2(a) and (d) have region contours of a relatively good quality. Fig. 2(b) is missing a major portion of the edges, while Fig. 2(c) is mostly meaningless because the target is not moving.

The segmentation maps are given weights of 4, 1, 1, and 1 for Fig. 2(a), (b), (c), and (d), respectively. Fig. 8 shows the intermediate result after the mle stage. Fig. 9 shows the intermediate result after the contour smoothing stage. Note the existence of small and noncompact regions in the image plane. Most edge contours have been smoothed, although the contours are not as smooth as one intuitively expects because of the *finite-drift* constraint. Fig. 10 shows the result after asserting the constraints on region size and compactness. Small regions have been eliminated. Note that stronger edges in Fig. 9 usually survived the constraint satisfaction stage. The result may be sent into the curvature minimization stage to further

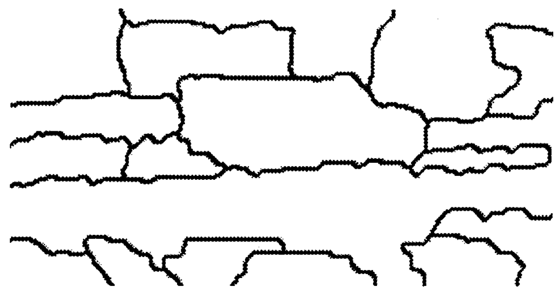


Fig. 10. The result after compactness and size constraints.

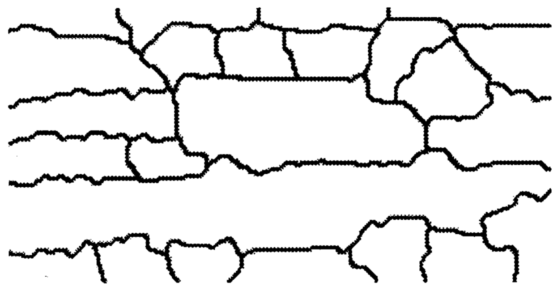


Fig. 11. The same weights as in Fig. 9, but $\lambda = 4$.

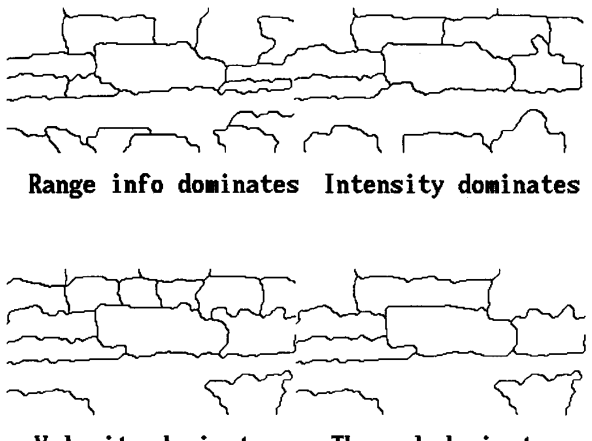


Fig. 12. Four integration results at $\lambda = 2$ that emphasize different inputs

smooth the edge contour. However, the resultant change is usually unnoticeable.

Fig. 11 shows the result of the integration using the same weighting factors but now with $\lambda = 4$. Fig. 12 shows four integration results (all with $\lambda = 2$). For the picture denoted as *intensity dominates*, the edge map derived from the ladar intensity image receives the strongest emphasis. The weights are now 1, 4, 1, and 1. Similar emphasis factors are applied to thermal and velocity segmentation maps. The result of emphasizing the velocity segmentation is particularly encouraging because the emphasized velocity segmentation misses the mark completely. The integration algorithm relies on information from the other three sources to derive the result.

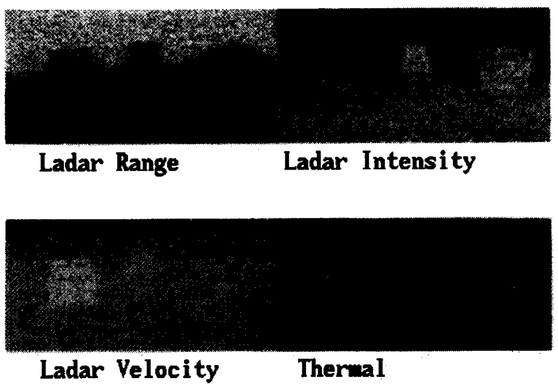


Fig. 13. Four source images for example 2.

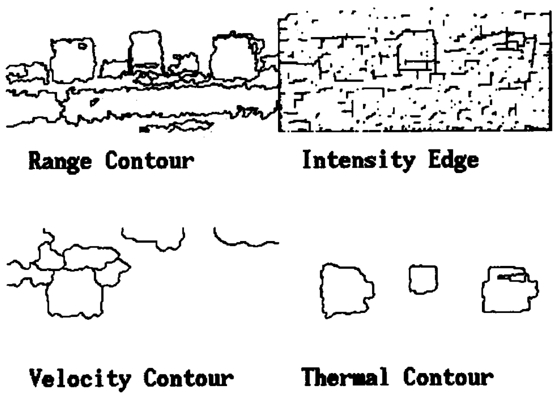


Fig. 14. Four input segmentation or edge maps.

Fig. 13 shows the four source images used in the second example. Two trucks are viewed from the tail end, with the left truck backing up toward the sensors and a bulletin board sitting between the two trucks. The inputs in Fig. 14 are obtained by the same methods as in the previous example. Fig. 15 shows the four integration results with $\lambda = 2$ and the same weighting factor as used in the first example.

The effects of different weights on input segmentation maps are consistent with expectations. All the integration results are similar while they have different control parameters and fusion radii. This demonstrates the robustness of the proposed algorithm. The integration results will further improve when *better* region/edge maps are used. If weights on individual information sources are given properly (for example, based on past evaluations of individual segmentation techniques), the result also improves. The CPU time for $\lambda = 2$ cases averages to 90 seconds, while the $\lambda = 4$ cases average to 130 seconds. From 20% to 30% of the CPU time is spent on the contour smoothing stage. Using a smaller ϵ significantly increases the CPU usage for contour smoothing but generates indistinguishable results.

Discussion

The very first issue in the estimation process is to determine *which* variable to estimate. It is difficult to estimate directly

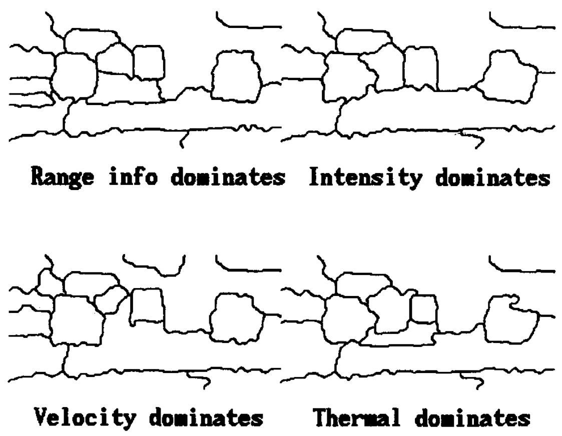


Fig. 15. The four integration results emphasizing different input components for Example 2 at $\lambda = 2$.

to which region an individual pixel belongs. Although one of the authors' previous work attempted a direct approach by intersecting two region maps, it could incorporate neither edge information nor user-supplied weighting [3]. One approach is to treat the entire image plane as a single array variable. The formulation might be terse, but the computation is formidable. Another approach is to estimate an array of the edge pixel positions as a vector variable. Because the number of edge pixels in the solution set is unknown, the size of the matrices cannot be fixed. In addition, the contour connectivity is difficult to represent directly in a matrix of pixel positions. Therefore, in this work, the estimation is aimed at attributes (position and strength) of edge pixels.

In many interpretation schemes, linear features and rectangular regions are important cues [24]. Additional constraints can all be implemented in the constraint satisfaction stage to preserve the desirable features. Of course, the merit function should be adjusted accordingly, and they can be much more complicated than the one in (9) for more delicate control. Nevertheless, the first two stages (estimation and smoothing) can remain unchanged and independent of the problem context. There will be cases where a simple merit function based solely on low-level information becomes unsatisfactory. However, this scenario is common in all bottom-up strategies that do not use high-level knowledge. One approach is to put the domain-dependent knowledge directly into the segmentation and integration modules, as in [7]. Another approach is to maintain the low-level modules as domain-independent as possible, while leaving the adaptation (to individual problem contexts) to the high-level modules through a feedback loop, as in [24].

This work differs significantly from that of Pavlidis and Liow [13] in several ways. First, their algorithm needs the original intensity images while ours does not. Second, the objective function in [13] is designed specifically to eliminate the artifacts generated by the quadtree-based segmentation algorithm. The segmentation algorithm is fixed, hence it restricts the design of the integration algorithm. In contrast, our algorithm is independent of the segmentation techniques. Third, it is not clear how user-specified weights and multiple segmentation maps from other techniques can be incorporated into their scheme. Finally, contour smoothing in their work requires repeated tracing through edge contours sequentially. No particular energy model for contour smoothing is considered in [13]. The focus of its smoothing stage is to stretch near-linear segments to linear segments where the intensity gradients are small. Higher order curves or strong edges remain intact (in [13, table II]). In contrast, our algorithm prefers compact region contours (subject to certain constraints).

Higher order curves, e.g., splines, may be used to link or to smooth edge contours, but more computation is needed. For example, one can fit a spline through several edge pixels to form an edge pattern. However, before the connectivity issue is resolved, it is hard to determine which pixel positions should be linked by the fitting curve. In comparison, each time the linear connection is enforced in the proposed algorithm, the length of the linear connection is guaranteed to be shorter than 2λ, while λ is usually small. (The examples in Section VII use 2 and 4 pixels for λ). Since the positions of the output edge pixels have to be quantized to integer coordinates, the potential benefit of using higher order contour interpolation for enforcing contour connectivity and the final smoothing is not significant.

The results in [16] are similar to those in [15] and both use the weighted average form for solution. The weights are determined by the standard deviations of observation noise in a generalized vector format, similar to that in equation (3). All test samples are used in [16] without considering the observation consistency or the fusion radius. Therefore, the question is still unresolved as which edge pixels from which input data set should participate in the estimation. Sher's result [17] is also in the form of a weighted average under certain assumptions (in [17, eq. (7)]) on input values, which are edge likelihood measurements. The weights are the probability for an input channel to make correct estimations. Reference [17] does not consider either figural continuity or contour smoothness. In contrast, our algorithm specifically considers continuity and does not use *a priori* information.

One interesting question is how much smoothing is necessary after the mle stage. For problems such as curvature estimation, which need contours with good analytical properties, [19] provides an analytical approach. It proposes an objective function for curve smoothing containing two terms (in [19, eq. (1)]):

$$\frac{1}{n}\sum_{i=1}^{n}(f(t_i)-y_i)^2+\mu\int_a^b(f^{(m)}(u))^2du. \qquad (10)$$

For the task of curve smoothing, our objective function for smoothing (equation (5)) is equivalent to changing the first term in (10) from the MSE criterion to the *finite-drift* constraint, and drops the term from the analytical expression. Therefore, it is unnecessary to find an optimal μ, as in [19]. This is a trade-off between computational efficiency and the level of accuracy of analytical curve properties needed for a particular task. The algorithm in [19] takes 40 seconds CPU time on a SUN-3/160 using 109 samples on a non-self-crossing contour, while the examples we show have 2200+ edge pixels in an edge map. The smoothing algorithm in [19] has to process noncrossing contour segments one by one, hence, there are potential difficulties at the contour junction points. Finally, all the curve smoothing results have to be quantized to integer coordinates. Therefore, minor differences in the floating-point representation of the pixel positions (hence the precision provided by an *optimal* smoothing) may become insignificant.

The smoothing mechanism in our work is similar to the work by Kass *et al.* in [22] (*Snake*) in the sense that curves are modeled as many individual (but connected) nodes. Iterative incremental changes are then exerted on these nodes. However, there is no real *fidelity* concern in *Snake*, while we have to smooth edge contours subject to the mle stage output. We do not consider *external forces* while it is an important component in *Snake*. The exception is when the *finite-drift* constraint or the *connected-neighbor* constraint is violated. If so, an infinite force is exerted to guarantee that these two *hard* constraints are satisfied. *Snake* needs edge tracing to set up an equation system correctly while ours does not. Our approach treats juncture points and nonjuncture points uniformly while *Snake* deals with only noncrossing contours. *Snake* also needs the original gray level images to compute the *force field*, and the force field needs to be updated at every iteration. For the smoothing task, our algorithm is likely to be more efficient. The centroid solution derived from our spring-node model satisfies *both* the α and β terms of in [22, eqs. (13) and (14)] by shrinking the length of the springs and reducing the roughness simultaneously. In our model, the balance of forces from neighboring nodes results in a simple local (weighted) average. In contrast, *Snake* has to solve a large (though sparse) equation system for every pixel at each step. Our system uses only local information and is equivalent to an iterative solution to the equation system in [22, eqs. (15) and (16)] by the *explicit method*. However, our approach does not need to choose a fixed step size γ artificially, as in [22]. Instead, the step size is determined jointly by the spring force and the two hard constraints.

VIII. Conclusion

We have presented an algorithm that integrates region and edge information to improve region segmentation and edge detection. The algorithm integrates segmentation information derived from different imaging sources and different processing techniques. It operates without intervention from high-level knowledge or the original gray-level images. User-specified weights and mixed types of input maps are allowed. The resultant contours are smoothed within the limit of channel resolution width. The proposed algorithm have been used as the basis of a multisensor image interpretation system [4], [5] using real images of outdoor scenes. The algorithm is efficient and the results are promising.

This work uses *maximum likelihood estimation* as the estimation strategy. Although *a priori* information is not considered, it is usually unavailable in practical situations. The algorithm addresses the issue of figural continuity and incorporates this concern into the solution procedure. During the initial estimation process, continuity influences which data points are considered in the weighted average operation. After the initial estimation, the *contour connectivity* constraint enforces connections between pixels that are originally connected. The curve smoothing stage uses the *spring-node* model to refine node positions iteratively, subject to the *finite-drift* and *connected-neighbor* constraints. The proposed algorithm treats juncture and nonjuncture pixels uniformly. Finally, the algorithm satisfies nonnegotiable constraints on region size and contour compactness if a region map is desired.

The proposed algorithm is not yet another segmentation algorithm. Instead, it is an integration algorithm capable of using all established segmentation algorithms as *front ends*. Many fast segmentation algorithms (but not necessarily very intelligent) may operate in parallel and then feed their results into the integration module. For example, multiple *shape-from-X* techniques can operate in parallel and then have their results integrated. Consequently, the need to design a single superior algorithm for segmentation is much less critical. Since only local information is used in the initial estimation and in the contour smoothing stages, the algorithm may be ported to parallel/distributed computer hardware when the maximum operation speed is required.

Acknowledgment

The authors would like to thank the Army Electro-Optical and Night Vision Center for providing the test images.

Appendix

The basic mle process considers one variable at a time. It is an established result [20], [21] that the mle is the limiting case of the *maximum a posteriori* (MAP) estimation when the *a priori* information approaches zero. From the classical MAP equation, "*a priori* information being zero" could be interpreted as the *a priori* density being uniformly distributed. In our problem context, this is equivalent to "edge pixels can be anywhere." For cost functions and *a posteriori* density functions satisfying certain conditions, the MAP result (as well as a class of estimators) is the same as the MMSE result [20], [21].

The following demonstrates a simple case. Given a set of observations of $z_i = s + n_i$, the conditional probability distribution of $\mathbf{z}$ given s under a set of independent Gaussian noise $\mathbf{n}$ (zero-mean, variance σ_i^2) is

$$p(\mathbf{z}|s) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma_i} exp(-\frac{(z_i - s)^2}{2\sigma_i^2}). \tag{11}$$

When we maximize this expression, we take a logarithm first, then take a partial derivative with respect to s, and finally set the result to zero. Thus we arrive at

$$\frac{\partial lnp(\mathbf{z}|s)}{\partial s} = \sum_{i=1}^{m} \frac{(z_i - s)}{\sigma_i^2} = 0. \tag{12}$$

The solution is a weighted average of all the input samples as described in (3).

Obviously, observation samples with larger σ's receive less weights. Suppose that the estimations of the x and y components are independent, and that the observation noises are independent and Gaussian; then the mle solutions for x and y can be formulated and computed similarly assuming that $\sigma_{xi} = \sigma_{yi} = \sigma_i$ for all i. Equations (9) and (30) in [16] mean essentially the same. When the user specifies different weights for different observations of z_i's, this specification is equivalent to claiming *a priori* knowledge of the ratio between σ_i^2's. From another point of view, $1/\sigma_i^2$ is equivalent to the combined effects of w_i, the user-specified (subjective) weight, and e_i, the edge strength at the position P_i. A larger weight (or a stronger edge value) implies a smaller σ_i and a more precise observation on edge locations.

References

[1] C. W. Tong, S. K. Rogers, J. P. Mills, and M. K. Kabrisky, "Multisensor data fusion of laser radar and forward looking infrared (FLIR) for target segmentation and enhancement," *Proc. SPIE*, vol. 782, pp. 10-19, May 1987.

[2] B. Bhanu and R. D. Holben, "Model-based segmentation of FLIR images," *IEEE Trans. Aerospace Electron. Syst.*, vol. 26, no. 1, pp. 2-11, Jan. 1990.

[3] C. Chu, N. Nandhakumar, and J. K. Aggarwal, "Image segmentation using laser radar data," *Pattern Recognition*, vol. 23, no. 6, pp. 569-581, 1990.

[4] C. Chu and J. K. Aggarwal, "Interpretation of laser radar images by a knowledge-based system," *J. Machine Vision Applications*, vol. 4, pp. 145-163, 1991.

[5] C. Chu and J. K. Aggarwal, "Multi-sensor image interpretation using laser radar and thermal images," in *Proc. 7th Conf. Artificial Intell. Applications,* Miami Beach, FL, Feb. 24–28, 1991, pp. 190-196.

[6] C. Chu and J. K. Aggarwal, "The integration of region and edge-based segmentation," in *Proc. 3rd Int. Conf. Computer Vision,* Osaka, Japan, Dec. 4–7, 1990, pp. 117-120.

[7] M. D. Levine and A. M. Nazif, "Low level image segmentation: an expert system," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 5, pp. 555-577, Sept. 1985.

[8] P. Fua and A. J. Hanson, "Using generic geometric models for intelligent shape extraction," in *Proc. DARPA Image Understanding Workshop*, Los Angeles, CA, Feb. 1987, pp. 227-233.

[9] A. Toet, "Hierarchical image fusion," *J. Machine Vision Applications*, vol. 3, no. 1, pp. 1-11, 1990.

[10] R. R. Kohler, "Integrating non-semantic knowledge into image segmentation process," Univ. of Massachusetts, Amherst, COINS Tech. Rep. 84-04, 1984.

[11] A. C. Bovik, "On detecting edges in speckle imagery," *IEEE Trans. Acous. Speech., Signal Processing*, vol. 36, no. 10, pp. 1618-1627, Oct 1988.

[12] H. L. Anderson, R. Bajcsy, and M. Mintz, "A modular feedback system for image segmentation," Univ. of Pennsylvania, Philadelphia, GRASP Lab Tech. Rep. 110, June 1987.

[13] T. Pavlidis and Y. T. Liow, "Integrating region growing and edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 3, pp. 225-233, Mar. 1990.

[14] J. F. Haddon and J. F. Boyce, "Image segmentation by unifying region and boundary information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 3, pp. 929-948, Oct. 1990.

[15] R. C. Luo, M. Lin, and R. S. Scherp, "The issues and approaches of a robot multi-sensor integration," in *Proc. IEEE Robotics Automation Conf.,* Mar. 31-Apr. 3, 1987, pp. 1941-1946.

[16] Y. Nakamura and Y. Xu, "Geometrical fusion method for multi-sensor robotic systems," in *Proc. IEEE Robotics Automation Conf.,* Scottsdale, AZ, May 14–19, 1989, pp. 668-673.

[17] D. Sher, "Evidence combination using likelihood generators," in *Proc. DARPA Image Understanding Workshop,* Los Angeles, CA, Feb. 1987, pp. 655-662.

[18] D. Marr, *Vision* San Francisco, CA: W. H. Freeman, 1982, pp. 62-63.

[19] B. Shahraray and D. J. Anderson, "Optimal estimation of contour properties by cross-validated regularization," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 11, no. 6, pp. 600-610, June 1989.

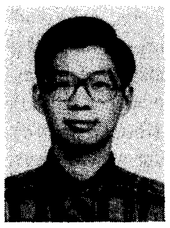
[20] M. D. Srinath and P. K. Rajasenkaran, *An Introduction to Statistical Signal Processing with Applications.* New York: Wiley, 1979.

[21] H. L. Van Trees, *Detection, Estimation, and Modulation Theory.* New York: Wiley, 1968.

[22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," in *Proc. 1st Int. Conf. Computer Vision,* London, England, June 8–11, 1987, pp. 259-268.

[23] C. G. Bachman, *Laser Radar Systems and Techniques.* Dedham, MA: Artech House, 1979.

[24] D. C. Baker, S. S. Hwang, and J. K. Aggarwal, "Detection and segmentation of man-made objects in outdoor scenes: concrete bridges," *J. Opt. Soc. Amer.*, vol. 6, no. 6, pp. 938-950, June 1989.



Chen-Chau Chu (S'88–M'91) graduated from the Department of Electrical Engineering of National Taiwan University, Taipei, Taiwan, in June 1981. He received the M.S.E.E. degree from the Department of Electrical and Computer Engineering of the University of Texas at Austin in July 1986, and the Ph.D. degree in May 1991.

He joined Schlumberger Austin Systems Center, Austin, TX, in June 1991 and now works on oilfield data interpretation. His research interests include computer vision, machine intelligence, computer graphics, and image/signal processing.

Dr. Chu was awarded the best student paper at the 1991 IEEE Conference on Artificial Intelligence Application.

J. K. Aggarwal (S'62–M'65–SM'74–F'76) has served on the faculty of the University of Texas at Austin College of Engineering since 1964 and is now the Cullen Professor of Electrical and Computer Engineering and the Director of the Computer and Vision Research Center. His research fields include computer vision, parallel processing of images, and pattern recognition. He is the author or editor of 6 books and 20 book chapters, and author of over 130 journal papers, as well as numerous proceedings papers and technical reports.

Dr. Aggarwal has been a Fellow of IEEE since 1976. He received the 1992 Senior Research Award of the American Society of Engineering Education. He is IEEE Computer Society representative to the International Association for Pattern Recognition, President of the International Association for Pattern Recognition, and Editor of IEEE TRANSACTION ON PARALLEL AND DISTRIBUTED SYSTEMS.